Homework 4

Phần 1: Lý thuyết phương pháp.

Ý tưởng của phương pháp chia để trị:

1. Divide: chia vấn đề thành các vấn đề con cùng kiểu nhưng có kích cỡ nhỏ hơn.
2. Conquer: xử lý các vấn đề con bằng sử dụng đệ quy.
3. Combine: kết hợp các kết quả của chương trình con vào kết quả của vấn đề ban đầu.

Lược đồ tổng quát của phương pháp chia để trị:

D&C (R) ≡

if ($R = R_0$)

Get solution D&C (R_0) ;

else

Partition R into R_1, R_2,..., R_n

for (i =1...n)

D&C (R_i)

Combine the solutions.

End.

Ví dụ: Tìm kiếm nhị phân trong 1 danh sách đã được sắp xếp.

Input: danh sách (a_1,...,a_n), x;

Output: vị trí i của ai = x, hoặc i = -1 nếu không có.

- **Ý tưởng:**

So sánh x với giá trị ở giữa của danh sách

Sử dụng kết quả để quyết định tiếp tục tìm bên nào của danh sách.

- **Lược đồ:**

```
BinarySearch(a,x, L,R):int ≡
        if (L=R) return (x=aL ? L : -1)
        else
                M = (L+R)/2;
                if (x = aM) return (M);
                else
                        if (x<aM) BinarySearch(a,x,L,M)
                        else BinarySearch(a,x,M+1,R)
                endif;
        endif;
End.
```

Phần 2: Bài tập tư duy

Exercise 1. Stock Pricing Problem.

- **Phân tích bài toán**
  - Input: danh sách $(a_1,...,a_n)$ giá cổ phiếu, $a_1 \neq a_n$ với mọi ngày.
  - Output: vị trí i của a_i sao cho a_i là ngày mua cổ phiếu và vị trí j của a_j sao cho a_j là ngày bán cổ phiếu sao cho đạt lợi nhuận tối đa

- **Lược đồ:**

```
buy_sell_stock(prices):int ≡
   n = len(prices)
        if n = 1
                return (0, 0)
        elif n = 2
                if prices[1] > prices[0]
                        return (0, 1)
```

```
        else
            return (0, 0)
    mid = n // 2
    left_buy, left_sell = buy_sell_stock(prices[:mid])
    right_buy, right_sell = buy_sell_stock(prices[mid:])
    for i in range(mid)
        if prices[i] < min_price
            min_price = prices[i]
        if prices[i] - min_price > max_profit
            max_profit = prices[i] - min_price
            left_buy = i
            left_sell = mid + i
    max_price = prices[mid]
    max_profit_right = 0
    for i in range (mid, n)
        if prices[i] > max_price
            max_price = prices[i]
         if max_price - prices[i] > max_profit_right
            max_profit_right = max_price - prices[i]
            right_buy = i - mid
            right_sell = i
    if max_profit_right > max_profit
        return right_buy, right_sell
    else
        return left_buy, left_sell
End.
```

Thuật toán: $T(n) = 2T(n/2) + n$

=> Có: $a = 2, b = 2, d = 1$

=> $a = b^d$

=> độ phức tạp của thuật toán là: O(nlogn)

Exercise2 : Unbreakable laptops

- **Phân tích bài toán**
  + Input: 2 laptops, tòa nhà n tầng.
  + Output: vị trí i của a_i sao cho a_i là tầng cao nhất mà thả laptop rơi xuống mà không vỡ.

- **Lược đồ:**

unbreakable(low, high, tries = 0):int ≡

if (low >= high) return low

middle = (low + high) / 2;

if drop_laptop(middle) breaks

return unbreakable(low, mid -1, tries + 1)

else

return unbreakable(mid + 1, high, tries + 1)

End.

Thuật toán: $T(n) = T(n /2) + 1$

Có: $a = 1, b = 2, d = 0$

=> $a = b^d$

=> độ phức tạp của thuật toán là O(logn)

Phần 3: Bài tập lập trình *(trong thư mục part 3)*

Phần 4: Đặt bài toán, thiết kế, phân tích và triển khai thuật toán

1. **Bài toán tìm đồng xu giả (Fake-coin problem)**
   - **Phân tích bài toán**:

     Input: danh sách n đồng xu, và có 1 đồng xu là giả.

     Output: vị trí của đồng xu giả.

   - **Xây dựng huật toán:**
     1. Chia tập đồng xu thành 2 phần bằng nhau.
     2. Đo trọng lượng của mỗi phần.
     3. Nếu trọng lượng 2 phần bằng nhau thì đồng xu giả không nằm trong 2 phần đó. Ta có thể tiếp tục tìm kiếm trong nửa còn lại của tập đồng xu.
     4. Nếu trọng lượng 2 phần khác nhau thì đồng xu giả nằm trong phần có trọng lượng nhỏ hơn. Ta tiếp tục chia phần đó thành 2 phần bằng nhau và lặp lại quá trình cho đến khi tìm được đồng xu giả.
   - **Phân tích thuật toán:**

     $T(n) = T(n/2) + 1$

     => độ phức tạp thời gian O(logn)
   - **Chương trình minh họa *(trong thư mục part 4)***

2. **Bài toán tìm cặp điểm gần nhất (Closest pair of point)**
   - **Phân tích bài toán:**

     Input: tập hợp các điểm $(a_1,...,a_n)$

     Output: 1 cặp điểm (a_i, a_j) sao cho khoảng cách a_i đến a_j là min trong tập $(a_1,...,a_n)$
   - **Xây dựng thuật toán:**
     1. Chia tập hợp các điểm thành hai phần bằng cách chọn một đường thẳng trực giao với trục tung. Điểm chia tập hợp là điểm nằm trên

đường thẳng này và có tọa độ trên trục tung bằng trung bình của tọa độ trên trục tung của các điểm.

2. Giải quyết bài toán tìm cặp điểm gần nhất trên hai tập hợp điểm con, sử dụng đệ quy.
3. Tìm khoảng cách nhỏ nhất giữa một điểm ở phía trái và một điểm ở phía phải, xét trường hợp nếu khoảng cách nhỏ hơn khoảng cách tìm được trong các bước trước đó thì cập nhật giá trị này.
4. Trả về cặp điểm có khoảng cách nhỏ nhất tìm được.

- **Phân tích thuật toán**

$$
\begin{aligned}
T(n) &= T(n/2) + n\log n + n \\
&= T(n/4) + n/2\log(n/2) + n/2 + n\log n + n \\
&= T(n/8) + n/4\log(n/4) + n/4 + n/2\log(n/2) + n/2 + n\log n + n \\
&= \dots \\
&= T(1) + n/2 n\log n \\
&= O(n\log n)
\end{aligned}
$$

=> độ phức tạp thời gian của thuật toán là O(nlogn).

- **Chương trình minh họa *(trong thư mục part 4)***